AD NUMBER

AD415721

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited. Document partially illegible.

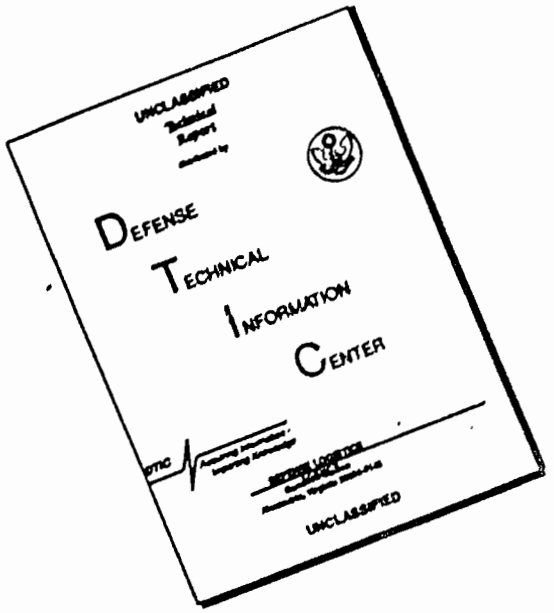
FROM:

**Distribution authorized to U.S. Gov't. agencies and their contractors;
Administrative/Operational Use; MAY 1963. Other requests shall be referred to Defense Research and Engineering, Washington, DC 20301. Document partially illegible.**

AUTHORITY

DDRE per DTIC form 55

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

63-4-6

SDC

TM-903 000/02

MANAGEMENT OF COMPUTER PROGRAMMING FOR COMMAND AND CONTROL SYSTEMS

8 May 1963

TM-903/000/02

TECHNICAL MEMORANDUM

(TM Series)

This document was produced by SDC in performance of contract SD-97

MANAGEMENT OF COMPUTER PROGRAMMING
FOR
COMMAND AND CONTROL SYSTEMS
A SURVEY

By

K. Heinze
N. Claussen
V. LaBolle

8 May 1963

SYSTEM DEVELOPMENT CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

A B S T R A C T

Programming managers, the Department of Defense, and the Air Force Systems Command are becoming increasingly concerned with improving programming and reducing costs. Thus, a study of management techniques for computer programming in command and control systems is being conducted at the System Development Corporation. In this document, the Computer Program Implementation Process (CPIP) project reports the findings of this study. Seven computer programming efforts were surveyed to reveal the characteristics of computer programming, including growth, military use, and staffing. In a section titled "Program Development Problems," such management difficulties as computer unavailability and the effects of customer delay in design decisions are discussed. The CPIP project recommends improving: (1) the identification of programming activities and products by, among other means, establishing a common technical language and, (2) customer relations, by recognition and understanding of customer needs (including his working environment), and the need for the customer to understand the developer's approach to the program. Managers have difficulty in controlling and planning programming efforts without precise and detailed cost data, standard performance measures, and definitions of tasks and products. Knowledge of managing and developing computer programming systems must be extended and detailed, and programming must be formalized.

PREFACE

In military applications of automatic data processing such as command and control, a computer usually performs many diverse tasks. Large, complex computer programs direct the performance of these tasks. Managers are showing a growing concern about the cost of these computer programs. Some managers use such crude rules of thumb as - the cost of programs will approximately equal the cost of the computer. Managers desire more objective measures of the programming cost/value relationships. During recent years, program developers have recognized these needs and have been examining programming job approaches.*

Management's growing concern with economic aspects of programming is shared by the Department of Defense (DOD), the customer for large computer programs used in command and control. DOD's Director of Defense Research and Engineering (DDR&E) has established a policy of reviewing the cost of research and development efforts. Apparently, computer programs have been identified specifically by DDR&E as a cost element to be monitored in command and control system development. Similar interest in computer-program costs is being manifested by the Air Force Systems Command, which has initiated studies whose goals are to reduce the costs of computer-program production.

A fundamental assumption of these studies is that costs will be reduced if management techniques for implementing large computer programs are improved. Consequently, a study of management techniques for computer programming is

*L. Christie, S. Fliege, J. Singleton, System Development of Command and Control Systems, (SP-182), SDC, October 1960.

C. M. Lawson, Computing Facility Management Survey Results and Analysis, (TM-704), SDC, March 1962.

T. Holdiman, "Management Techniques for Real-Time Computer Programming," ACM, July 1962.

W. Hosier, "Pitfalls and Safeguards in Real-Time Digital Systems with Emphasis on Programming," IRE Engineering Management, June 1961.

Lt. Col. R. V. Jordan, "Programming Talent Can Be Measured," Datamation, July 1962.

T. Edward Bissell, "Measuring Programmers Effectiveness," Guide to Data Processing Personnel Management, American Data Processing Inc., 1961.

R. Patrick, "Let's Measure Our Own Performance," Datamation, June 1962.

L. Schultz, Proceedings of the Symposium on Information Processing in Command and Control Systems, (TR-4), SDC, June 1962.

being conducted at the System Development Corporation as part of the command research sponsored by the Advanced Research Projects Agency (ARPA). This study, called the Computer Program Implementation Process (CPIP) project, is attempting to relate the computer programming process to its products and identify techniques that will reduce costs and lead times and yield improved products. The results of the study will appear in a guidebook for management authored by the members of the project.

The project's early efforts focused on (1) defining and analyzing computer-programming products, activities, tasks, and needed resources, and (2) gathering data about the experience of other developers of computer programs. CPIP project members interviewed managers, studied budget statements, examined results of similar studies, and reviewed technical documents describing computer programs in the course of surveying seven large programming efforts. This document summarizes the results of that survey.

Project members* had difficulty in gathering accurate and complete numerical data; consequently, though the numerical data presented do provide some insight into the programming process and establish some gross cost relations, these data may be regarded as interim findings that will be refined in future reports.

*Project members, other than the authors, who participated in the survey analysis are L. Farr and A. Rosenberg.

(page 4 blank)

CONTENTS

ILLUSTRATIONS

Figures

Tables

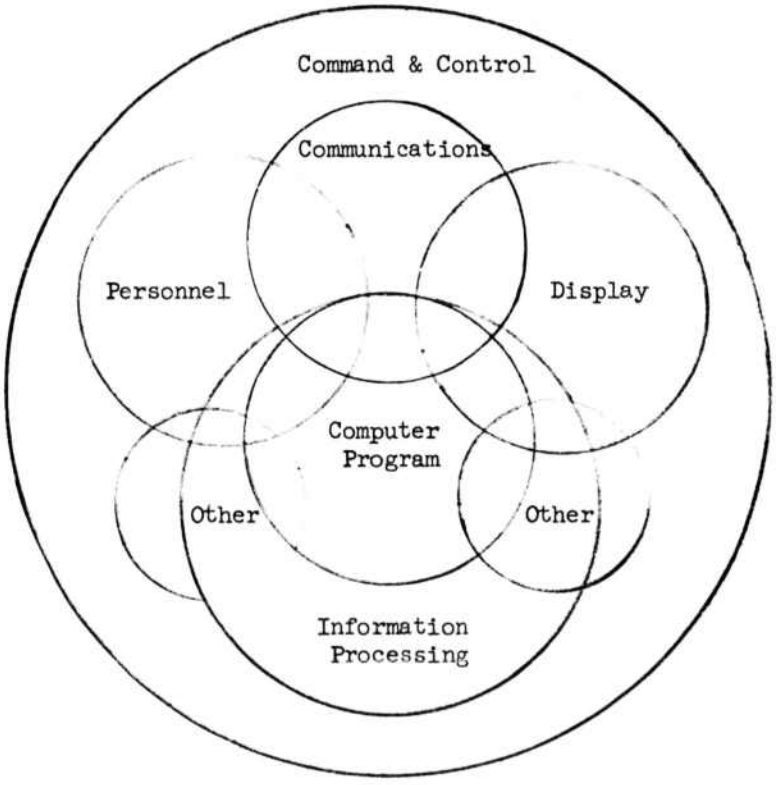


FIGURE 1. INTERRELATIONSHIP OF THE COMPUTER PROGRAM SYSTEM WITH OTHER SYSTEMS FORMING THE COMMAND AND CONTROL SYSTEM

1.0 INTRODUCTION

This report surveys the computer-programming efforts of seven large command and control systems. Section 2 presents an overview of computer programming--the activities, characteristics, development for military systems, staffing, growth of specialization, etc. Section 3 presents a set of representative problems that are encountered in managing a large computer-program development effort. Section 4 represents CPIP "Opinions and Recommendations" for the solution of the problems recounted in Section 3. Section 5 contains a series of graphs and a chart that permits quantitative comparisons (man-months, program instructions, computer hours, etc.) of the costs of the seven systems surveyed.

2.0 COMPUTER PROGRAMMING

This section describes computer program development and shows how the peculiarities of the task affect its management.

2.1 Development

The subsystems constituting the command and control system are so closely interrelated that it is difficult to separate one from another. Figure 1 shows the pronounced dependence of the computer program on the other subsystems and components within the command and control system.

Figure 2, "Computer Program System Development," shows an idealized work flow and lists the major activities. The CPIP survey is concerned primarily with the management of these major activities. Figure 2 shows how the developer assists the customer in the initial phases of program development, i.e., by defining the system requirements in the precise detail needed for computer programming.

2.2 Growth

Originally, a small group, usually associated with the development of electronic digital computers, performed all programming activities. Since then, the design, production, and maintenance of computer programs have become specialized activities that require many people to perform only one of them; e.g., program test; however, there is a continuing and vital need for programmers with broad experience--generalists who thoroughly understand programming in its various applications; who can, for example, judge various approaches to program design and evaluate tradeoffs such as machine running time versus storage allocation.

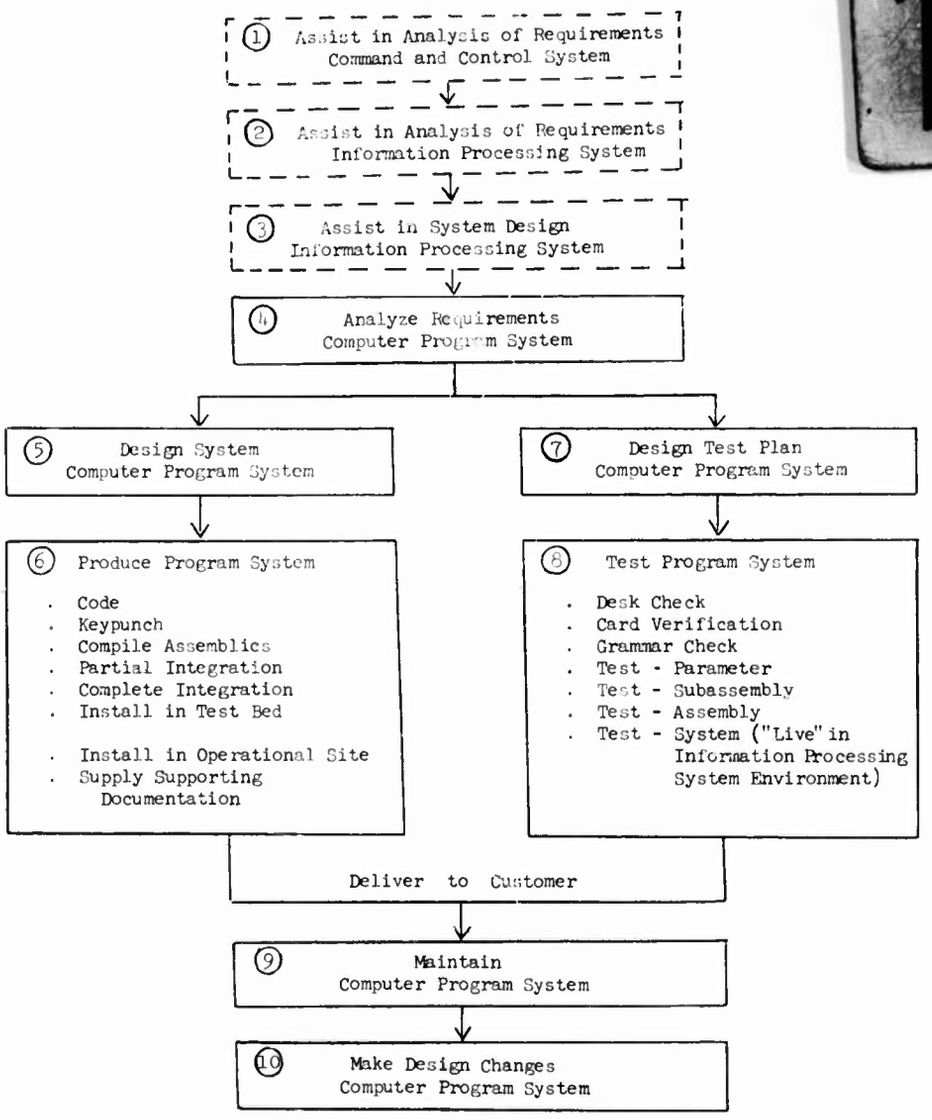
2.3 Military Use

Military needs often dictate that developers of large complex programs work to a tight schedule. In this situation, the program developer must compartmentalize his work, and must coordinate not only activities within his own organization

FIGURE 2.

COMPUTER PROGRAM SYSTEM DEVELOPME

WORK FLOW - A SCHEMATIC MODEL



Legend:

1. Numbers in boxes refer to activities itemized in List of Activities.
2. Boxes numbered 1, 2, and 3 (with broken lines) imply that the program developer assists the user or customer, as requested, in performing that activity or task.
3. After acceptance of the computer program by the customer, the system characteristically becomes operational, but the development of the system continues in the form of operating improvements, retrofits, etc.--all of which usually proceed through the development cycle charted above.

The dev
below.
Command

Command

(1) As
th
Sy
in

Informa

(2) As
in
ma

(3) (b
Pr

Compute

(4) An
pr

(5) Sp
pr
dc

(6) Pr

(7) De
th

(8) Te
as
en
(S

(9) Ma
af

(10) Ma
su

Program

Te

Pr
Op
Co

Pu

Co

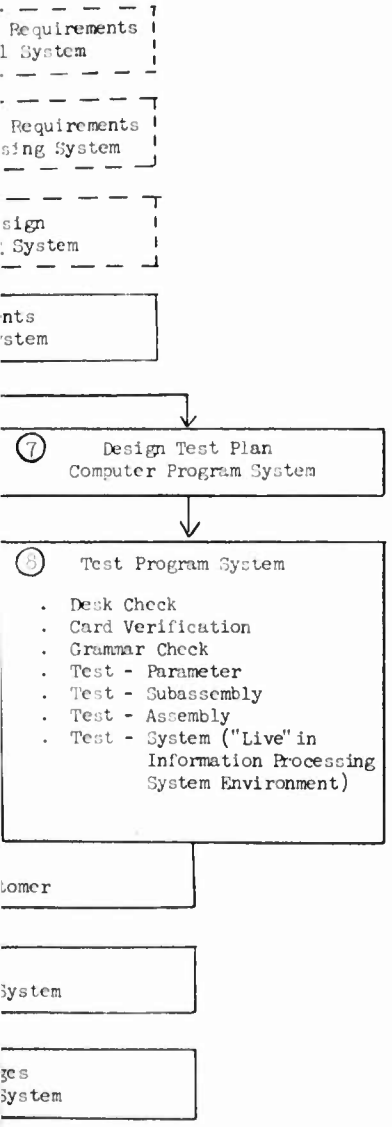
Adr

Di

FIGURE 2.

COMPUTER PROGRAM SYSTEM DEVELOPMENT

IC MODEL



s itemized in List of Activities.

oken lines) imply that the program
ner, as requested, in performing

gram by the customer, the system
al, but the development of the
ating improvements, retrofits,
hrough the development cycle

LIST OF ACTIVITIES

The developer of the computer program system performs the activities itemized below. He also assists the customer or user in performing activities of the Command and Control and the Information Processing Systems ((1), (2), (3)).

Command and Control System

(1) Assist the user or customer in analyzing the user's requirements (primarily those for the Information Processing Subsystem) of the Command and Control System (i.e., developer helps translate requirements from customer's language into data-processing language).

Information Processing System

(2) Assist the user or customer in (a) determining the requirements of the information-processing job (i.e., the allocation of tasks among men and machines and the procedures for handling the data) and

(3) (b) specifying the design of these tasks constituting the Information Processing System in operational specifications or operating system descriptions.

Computer Program System

(4) Analyze program system requirements to establish the gross structure of the program system(s).

(5) Specify the computer program system design (i.e., determine the data to be processed and its timing constraints; determine the computer storage needs; design data formats or message formats, interprogram communications).

(6) Produce (code) the machine instructions for the program system.

(7) Design Test Plan (i.e., for verifying all computer programs--subprograms through complete program system).

(8) Test individual subprograms (parameter test), collections of subprograms (sub-assembly test), entire program system in a simulated or partially "live" environment (Assembly Test); test program system in operational environment (System Test).

Note: As necessary, program developer may install and test computer program system as the operational model in a computer or a site different from the one used for development.

(9) Maintain the computer program system (i.e., detect and correct errors found after the system goes operational.)

(10) Make design changes aimed at improved performance, and incorporate approved suggestions by modifying operating program and procedures.

Programming Support Activities

Technical

Prepare (design, code and test) support programs such as utility, facility, etc.
Operate a computer facility.
Collect data (a data base) that is to be stored in the computer (e.g., aircraft flight characteristics).
Publish and maintain contract-required documentation such as operational and program design specifications.
Collect information concerning operating conventions and procedures to facilitate understanding with user.

Administrative (Management)

Direct, plan and coordinate all of the above activities.

but also interfacing activities with other component developers. Further, the program developer must acknowledge that projected military missions and operations fluctuate with funding, which, in turn, affects equipment and program design. Finally, since military command-control systems are usually designed for a single command at a single location, the designs and the development process usually do not incorporate improvements that might evolve from repetitive production or the use of standard modules.

2.4 Personnel

New uses for automatic data processing and specialization and growth of computer programming have increased the demand for specialized personnel; however, the demand still exceeds the supply. Opportunities are plentiful for personnel to secure jobs with increased salary and/or challenge, making it difficult for programming managers to keep their experienced people. This lack of continuity occurs in the Services too with the periodic rotation of military personnel. Some military commands have recognized this problem, however, and have increased the length of tours of duty.

2.5 Computer Programming Characteristics

The following are representative computer-programming characteristics. To some extent, these characteristics define computer programming in command-control system development and show the problems with which technical management must cope.

- The computer program, the basic product of programming, is intangible and has many interdependent parts.
- Program design is highly dependent upon the design of other system components and operating procedures in the command.
- Program design changes continually.
- Program design documentation must be under continuous and active control.
- Documentation specifying program products must be readily available, ample and timely.
- Program developers and users must coordinate.
- High personnel turnover impairs work continuity.

3.0 MANAGEMENT PROBLEMS IN PROGRAM DEVELOPMENT

This section discusses briefly management problems that occur in computer-program development; the problems concern:

- Equipment--the computer and its supporting components, upon which the program depends; and

- Customer relations--dealings with other developers and with the user, concerning program development.

3.1 Equipment Problems

- Unavailability of the computer. Failure to deliver or check out a computer on schedule delays programming; competition for the computer with other users slows response time. In both cases, cost increases (for rental of another computer and programmer travel time).

- Unreliable computer. An unreliable machine may waste a programmer's time in verifying the correctness of a program or delay his testing if it is down for unscheduled maintenance.

- Changes in equipment. Changes in equipment design during program development usually require change in both operational design and program design, and may require repetition of coding and testing. Further, when equipment subsystems (other than the computer and peripheral equipment subsystems) are developed concurrently with the computer programs, it is difficult to obtain up-to-date design information about the developing operational system. Correcting "errors" resulting from obsolete information necessitates modifying the design and probably more program and system testing. Also, if participants in system-test planning and execution are unfamiliar with the reliability of the equipment or the measures used to describe it, the evaluation of the program (and, in turn, of the entire system) during a test will be slowed.

- Inaccurate documentation of computer design or accompanying "software" programs. If the manufacturer's specification of a computer's operation or of its support programs is obsolete or inaccurate, the program design will be incompatible with the computer or the program instructions will be inoperable. This problem tends to occur when the computer and program are developed concurrently.

- Unreliable programs supplied by manufacturer or user-groups. Support programs insufficiently tested by the computer manufacturer become a source of errors for the program system developer, confusing checkout and lengthening development time.

- Use of a substitute computer for program test. When programs must be tested on a computer different from that used in operation, the results may not be entirely realistic and complete, and later testing with the operational computer may be required.

Inadequate computer size. Economizing on computer size, especially not allowing for growth of memory, is false economy; it may require modification of program design if (as has often been the case) delivered programs are substantially larger than early estimates.

3.2 Customer-Relations Problems

Misunderstandings may occur between program developers and customers. The customer may not understand the magnitude of system development---the role of automatic data processing and the detailed design requiring his participation and approval. Conversely, the developer may not understand the customer's requirements, military missions, and constraints. Further, there is apt to be a language barrier: the customer uses military terms; the program developer uses the language of automatic data processing. Similar misunderstandings arise between program and other component developers.

Some specific problems are:

- Lack of a well-defined management concept. When the customer lacks a well-defined management plan for developing the system, the boundaries of job responsibilities among contractors cannot be clearly specified. Such vagueness results in sluggish development for parts of the system.

- Absence of channels of communication. The absence of well-defined, approved channels of communication between developer and customer (e.g., for negotiating schedules) hinders the program because it leads to misleading information. This problem occurs most frequently when many representatives of both customer and developer are involved.

- Customer delay in design decisions. The customer's indecisiveness in acting on specifications submitted to him may lead to uncertainty, time loss, and costly repetition of the program-design work that usually continues while he delays.

- Change in requirements. The customer may make changes in requirements during program development and still expect previously agreed-upon schedules and budgets to remain in effect. Such changes, however, depending upon their magnitude and the time at which they are introduced during the program development process, may require (1) extensive changes to operational and program design, and (2) repetition of tests accompanied by new test planning and design. The program developer, on the other hand, cannot easily predict the impact and cost of a change in requirements.

  Since change is inevitable in the development of a large system, both customer and program developer are frustrated by the dearth of tools for budget and schedule control.

- Discontinuity of understanding. Both the customer and the developer have personnel changes during program development. Such changes impair understandings acquired over many months of transacting the business of program development.

4.0 OPINIONS AND RECOMMENDATIONS

This section presents two categories of recommendations intended to ameliorate or solve the problems discussed in Section 3. The first category concerns rudimentory techniques that deserve further development; the second, the need for research to extend current technology.

4.1 Improve the Identification of the Activities and Products of Program Development

a. Establish a "common language." Technical management needs a standard vocabulary to supplement the glossary being developed for automatic data processing. The vocabulary could be revised periodically to keep up with the state of the art and to augment and refine existing terms in the vocabulary. So that costs can be effectively related to work and to products, these terms should be consistent with existing management-reporting frameworks, e. g., PERT used on command and control system development.

b. Publish work plans. The program developer should publish and maintain current, detailed work plans. In the dynamic environment of program development, such plans serve to coordinate the work within the program developer's organization, pinpoint the impact of proposed changes, and inform other developers and the customer on progress and responsibilities.

c. Improve documentation. There is a continuing need to improve documentation in program development. Correct and complete documentation provides a record of milestones, which assists in management control, and helps to maintain work continuity.

Documentation should be recognized as a major activity in the program development effort. From the early phases of program development when, for example, the documentation translates user requirements into data-processing language to delivery of the completed program system documentation should be managed and controlled efficiently. With a clear view of both the customer's and his own requirements, the program developer should establish controls for document production, quality, and retrieval.

Production control includes listing documents to be produced; assigning responsibility; selecting the production method; scheduling, monitoring; progress reporting; and incorporating and authenticating changes.

Quality control includes establishing quality standards and tolerances for editorial and technical content, instituting procedures to assure that quality standards are met; and establishing standard formats, content requirements, and communication channels.

Information retrieval. Resources should be allocated and techniques selected or devised to accomplish effective document maintenance and efficient information retrieval. Program design control, for example, would be improved by numbering an entire set of documents in a standard manner that would permit matching the computer program description, as expressed in words and drawings, with the resulting program represented by card decks and listings.

d. Collect more detailed cost and product data. Currently, most cost data collected concerns the budget status of organizations and contracts; few cost data are related to the products of programming. Consequently, it is difficult to estimate confidently the cost of programming activities and products and to compare alternate techniques of doing a job.

Although acquiring additional cost information would initially add to programming costs, the knowledge gained about the relationships among activities, products, and resources would permit more efficient production and better decisions, and, eventually, reduce the over-all cost of computer programming. (For a list of factors affecting program cost see "Cost Factors for Estimating the Development of Large Program Systems" in the Appendix.)

e. Improve tools for accommodating program changes. Since change is a normal characteristic of military computer programming (requiring at least half the programming time), the developer should seek improved techniques to:

(1) Identify all parts of the program and documents affected by proposed changes;

(2) Standardize the evaluation of changes;

(3) Estimate the costs of changes (in manpower, computer time, and impact on schedules);

(4) Process, test, and install changes;

(5) Modularize subprograms so that changing them will have minimum impact on other subprograms; and

(6) Increase the responsiveness of programs to change, by designing them to use "parameter ranges" rather than single values as a basis for design, allowing growth with few changes in instructions.

f. Conduct research to define standards for programs, tools to produce programs, supporting documents, etc. Running time and computer storage requirements are examples of existing measures of program performance. Program modularity, changeability, and usability are other features that should be examined and defined more precisely.

g. Validate the product. During computer program development periodic validation of program products is necessary for both developer and customer--the developer needs it because of compartmentalization, division of labor, and geographical separation of personnel; the customer requires it because of the essentially intangible nature of the program. To some extent, validation occurs at critical points in operational design, e.g., customer concurrence on specifications, but beyond this, validation is inadequate. Test results are seldom recorded and no standard vocabulary is available for communicating these results.

Intermediate validations would help build confidence in subprograms. Each subprogram would have a label or synopsis describing previous test results for the programmer who is testing a change. Upon completion of the program, the developer would provide the customer with a "certified test summary" giving, for example:

- Number of hours of error-free program-system test under live-environment conditions
- A curve of the number of detected and corrected errors plotted against time (from a specified series of program system tests)
- The range of inputs tested during the entire program and some indication of the response obtained
- The illegalities tested
- System reactions (e.g., running time) under various input loads
- Results of tests of major branches (e.g., priority interrupts).

The program developer could use similar devices to show the accuracy of documentation, both during program development and upon its completion.

Such a validation system would (1) sharpen systematic detection and correction of errors in the program system, (2) build confidence in changed programs, and (3) allow the developer to summarize the elements of validation for customer approval. Such statements would, of course, improve the developer's ability to compare what the program actually does with what it should do.

h. Formalize and extend present techniques of programmer training. The following are examples of important programmer activities that should be formalized and taught:

(1) Documentation techniques and style formats for all programming

(2) New design techniques for data storage and retrieval, executive control, and economical computer use (e.g., storage, running time)

(3) Analysis, synthesis, and formulation of operational system descriptions

(4) Design and analysis of program test tools and diagnostic techniques.

4.2 Improve Customer Relations and Understanding of Programming Environment.

For the developer to understand the customer and his working environment, and for the customer to appreciate the developer's approaches to programming problems, the developer must:

a. Assess the customer's working environment and publish the findings.

Before, or soon after, beginning work on a new contract, the developer should select a team of senior personnel to analyze the customer and his working environment. Such an analysis, distributed to appropriate supervisory and technical staff and kept current, could (1) improve detailed planning, (2) indicate potential problems, and (3) serve for indoctrination. Presuming the customer is one of the services, the analysis should include:

- military mission
- organization of the military agencies involved
- functions within the organization
- key personnel and their responsibilities
- facilities
- what the customer expects of the new system
- what the customer expects of the developer
- computer installation and customer's knowledge of current capabilities and problems of data processing and information handling

(page 18 blank)

- a "first" plan for carrying out the computer program
- a bibliography of useful and available documents on subjects listed above.

b. Recognize that developing operational requirements is part of developing the program. In developing the computer program, programming personnel are actually detailing part of the total operational system design. Besides discovering requirements that are infeasible, programmers may discover new uses for automatic data processing, enabling them to extend and improve system requirements.

c. Recognize that instructing the customer is part of the over-all program. Educating the customer requires not only the many briefings on the process and problems of computer programming, but also lucid, formal documents. Such documents should emphasize concrete ideas and tangible products and make liberal use of graphical material.

Further, the developer must familiarize himself with the customer's day-to-day operation and language. With more mutual understanding, the separate skills of customer and developer will better mesh during system development.

d. Qualify the items and tools used in developing computer programs. The program developer should establish quality standards, using quantitative terms, where possible, for items such as computers and manufacturer-supplied programs, and have the customer relay these specifications to the supplier. The intent is, of course, to make the developer's expectations clear and precise and to avoid disappointments.

e. Use PERT as a tool for negotiating schedules and promoting concurrences. PERT can facilitate effective communication and negotiations in concurrence, scheduling, and coordination. To use PERT for these purposes, the program developer must have a network that records not only significant milestones (events or activities) for his own work, but also the dependencies on the major milestones in the entire military system (preceding and succeeding nonprogramming milestones for which the developer is not responsible.

Obviously, if milestone networks are used in this way, they must (1) be revised continually to reflect changes in plans as the system develops, and (2) show all supporting activities and decision points, instead of merely the principal work flow.

5.0 COMPARISON OF DATA

For the seven programs surveyed by the CPIP project, this section summarizes quantitative data on the core (i.e., design, code, test portion) of the programming job.

- The number of instructions in the computer programs, both operational and utility
- The number of man months* used for program design, production, and test activities
- The number of computer hours used
- The number of pages of contract-required documentation.

For anonymity, the contracts are denoted as "A" through "G."

Table 1 summarizes cost data; Figures 3 through 9 are graphs of various data;** Tables 2 through 8 are fact sheets listing, for each contract, significant characteristics of both the program and the development that should help the reader interpret the cost data presented in the graphs.

Although the charts provide some insight into programming costs and their relationship to product size, the data have limited use for planning a large programming effort. First, the data were assembled without benefit of a rigid set of rules and definitions. Second, even if a sufficient number of cases were available for high confidence in the data, other data, establishing the relation between requirements and number of instructions, are needed to permit a manager to make accurate cost estimates for programming. Ways must be sought to assign measures to requirements, such as complexity of the data-processing tasks, size of the data base, and operating-system response time. These measures, in turn, may be correlated with number of instructions to find which of them can be confidently used as predictors. Third, the variable "number of instructions" or "program size" cannot be used as an exclusive measure of the product. The popularity of size as a measure is based primarily on its availability. Other important program measures are needed: measures of quality, e.g., freedom from error; performance, e.g., program operation time; and convenience or "usability." Also, an obvious shortcoming of program size as a measure of

---

* Man-months, as used in this section, refers to effective man-months. An effective man-month is the time charged directly to a contract after sickness, vacation, holidays, etc., have been deducted. These-man month figures include lower levels of supervision, but do not include higher levels of management, usually allocated to a contract indirectly.

**One relationship of interest not depicted in the graphs is the ratio of operating program size to utility program size, which, if we disregard the record of contract "E," is roughly $1\frac{1}{2}$ to 1.

value is that clever, experienced programmers can generate the same logic with fewer instructions than those needed by inexperienced programmers. Even if programmer capability were a constant, the variation in logical power of various machines and their order codes would distort comparisons among efforts in terms of program size. Further, if memory storage space is at a premium, much programming effort will be used simply to reduce the number of instructions. Therefore, despite the availability of program size as a basis for comparing programs, dangers exist when it is used without qualification.

TABLE 1. SUMMARY COMPARISON OF COST DATA BY CONTRACT

CONTRACT	OPERATIONAL PROGRAMS			UTILITY PROGRAMS			RATIO OPS/UTIL	TOTAL PROGRAMS			COMP HOURS	RATIO INSTR PER COMP HR	PAGES OF DOCUMENTS
	INSTR	M/M	RATIO INST/MM	INSTR	M/M	RATIO INST/MM	INSTR	INSTR	M/M	RATIO INST/MM			
A	22,339	131	171	10,802	25	432	2.2	33,141	156	212	810	41	1,502
B	224,210	1426	157	149,230	244	612	1.3	373,440	1670	224	5506	68	5,000
C	71,400	624	114	(60,000)(1)	-	-	1.2	71,400	624	114	2986	24	1,800
D	88,654	300	295	38,236	60	637	1.1	126,890	360	352	1250	101	-
E	165,128	715	231	33,443	198	169	5.0	198,571	913	217	4166	48	-
F	415,000	(2)	-	225,000	(2)	-	(2)	640,000	6975(3)	92	14,500	44	(2)
G	286,000	3000	95	173,000	600	288	1.6	459,000	3600	128	10,400	45	5,550(4)

Legend:

(1) Already available from another contract

(2) Not available

(3) Incomplete

(4) Excludes 3,000 pages of site-adaptation data

NOTE: Number of instructions refers to machine-operating instructions, and does not include the data stored in external (to subprograms) data tables.

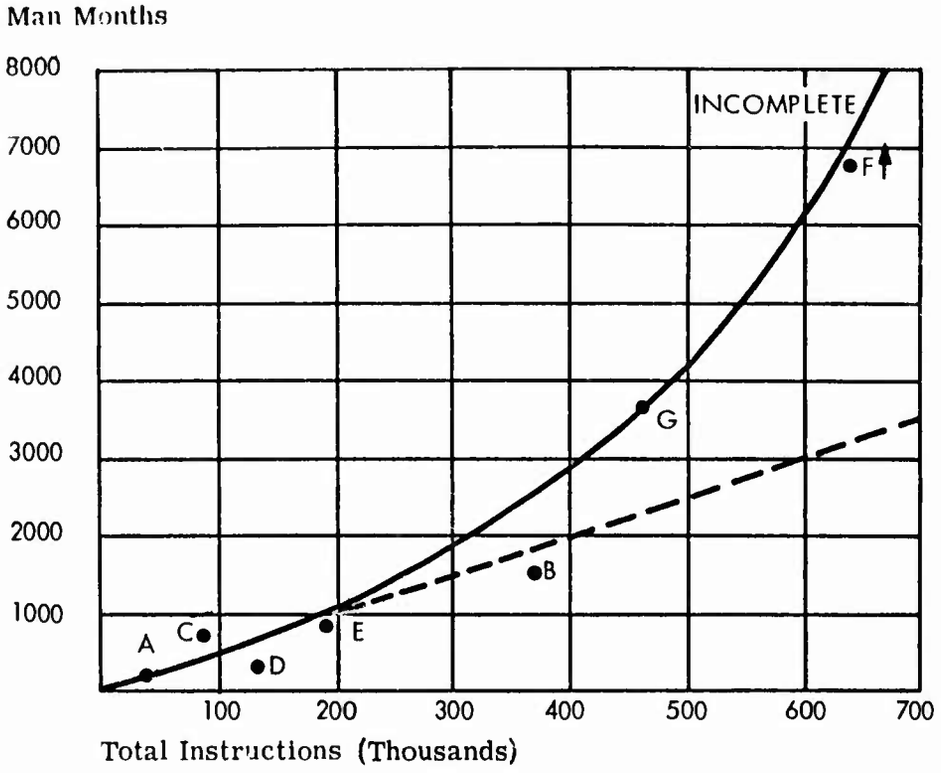


FIGURE 3. MAN MONTHS VERSUS PROGRAM SIZE
(PROGRAM DESIGN, CODE, & TEST)

This figure shows the costs of programming in man-months as a function of the total number of instructions written (for both operational and utility programs). Because the data are so gross, no attempt was made to calculate an estimating function. The curve is an "eyefit." For comparison, a dotted line representing a rate of 200 instructions per man month, a popular rule-of-thumb, has been inserted. In this range of program size, it appears that the cost of programming is an exponential or parabolic function of program size. This may be due to such factors as increases in communication and coordination and increased complexity in the larger, interrelated programs. The arrow in the upper right corner indicates the expected direction in which point F will move when the program effort represented by it is completed.

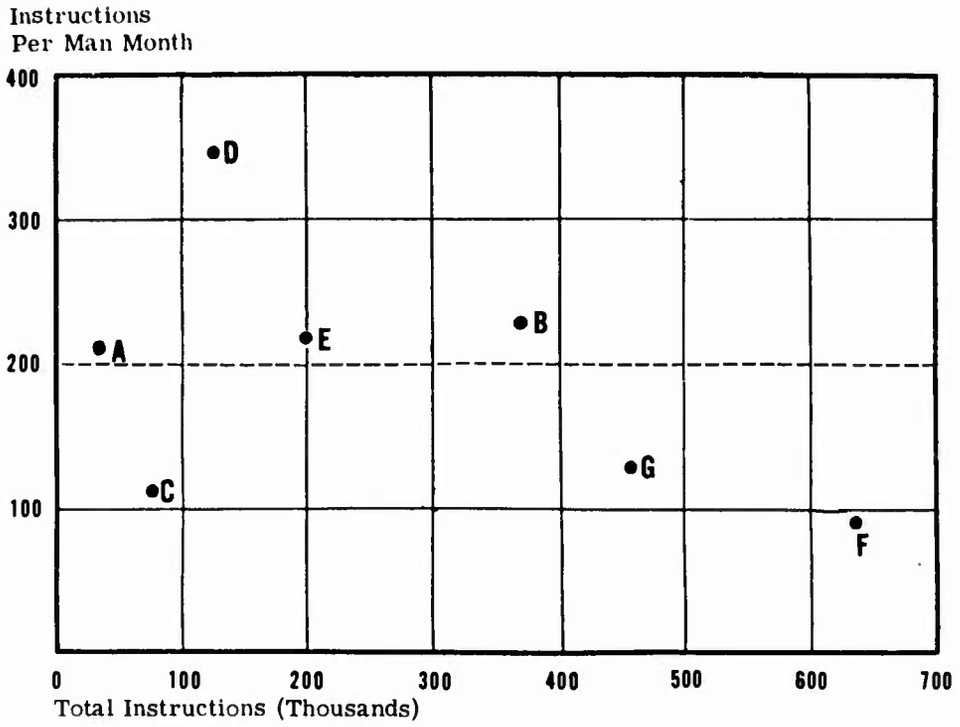


FIGURE 4. INSTRUCTIONS PER MAN MONTH VERSUS PROGRAM SIZE

In this scatter diagram a dotted line is shown at 200 instructions per man month, a popular rule-of-thumb used in making estimates. In general, the points reflect a decreased output per unit cost, i.e., number of instructions per man month, as the program size becomes larger. In work on smaller programs or program systems, e.g., less than 10,000 instructions, data show rates ranging from 400 to 1000 instructions per man month for individual programs.

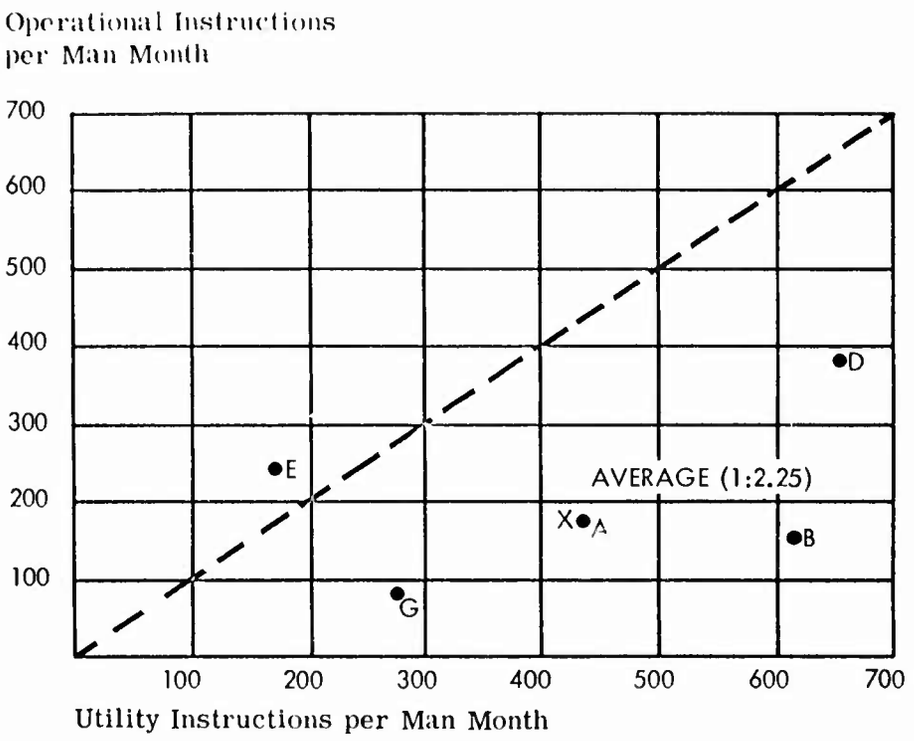


FIGURE 5. NUMBER OF OPERATIONAL INSTR./MAN MONTH VERSUS NUMBER OF UTILITY INSTR./MAN MONTH

Except for E, all points in this figure fall below the 45° (equal-eost) line suggesting that the cost of developing utility programs is less than that for operational programs of comparable size. The average ("X") of the five points is 1 operational to 2.25 utility-program instruetions per unit of manpower effort. One explanation for the lower cost may be that the program developer is his own customer for the utility program system; therefore, requirements are easier for him to define and stabilize, and little external coordination is required of him. One factor not considered here is the eost of machine time; this may be greater for initial utility programs, when no program tools are available to support tests.

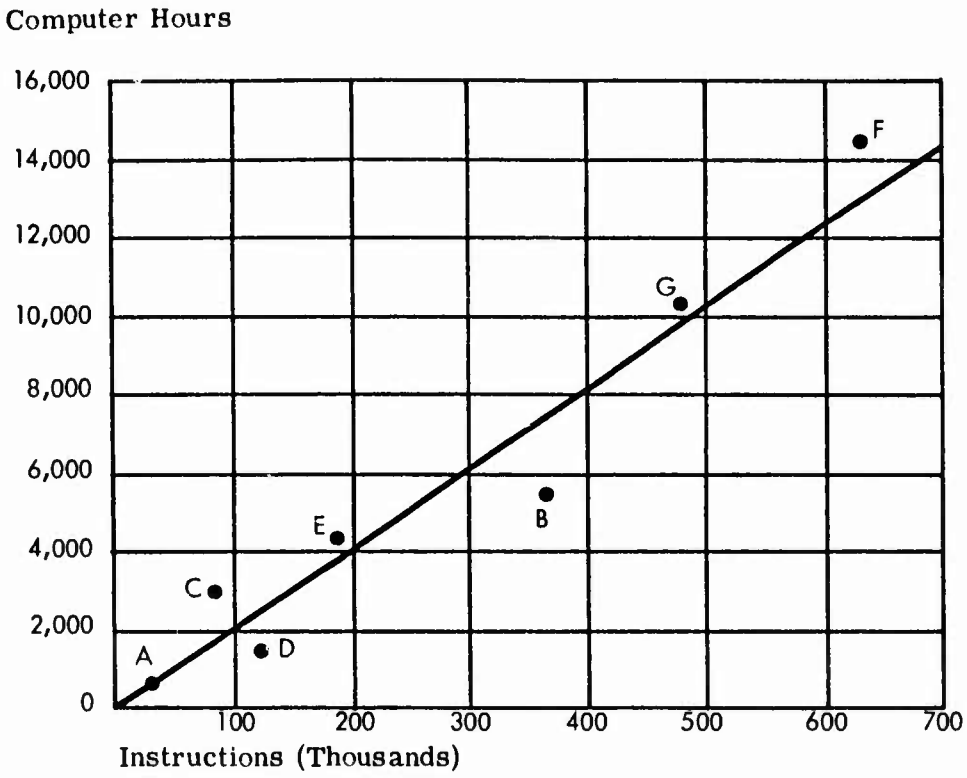


FIGURE 6. COMPUTER HOURS USED AS A FUNCTION OF PROGRAM SIZE

The growth of machine-time use with program size appears to be linear. Points B and D, representing programming efforts in which a procedure-oriented language is used, fall below the line. This suggests that use of a procedure-oriented language may reduce the amount of machine-time needed for program development.

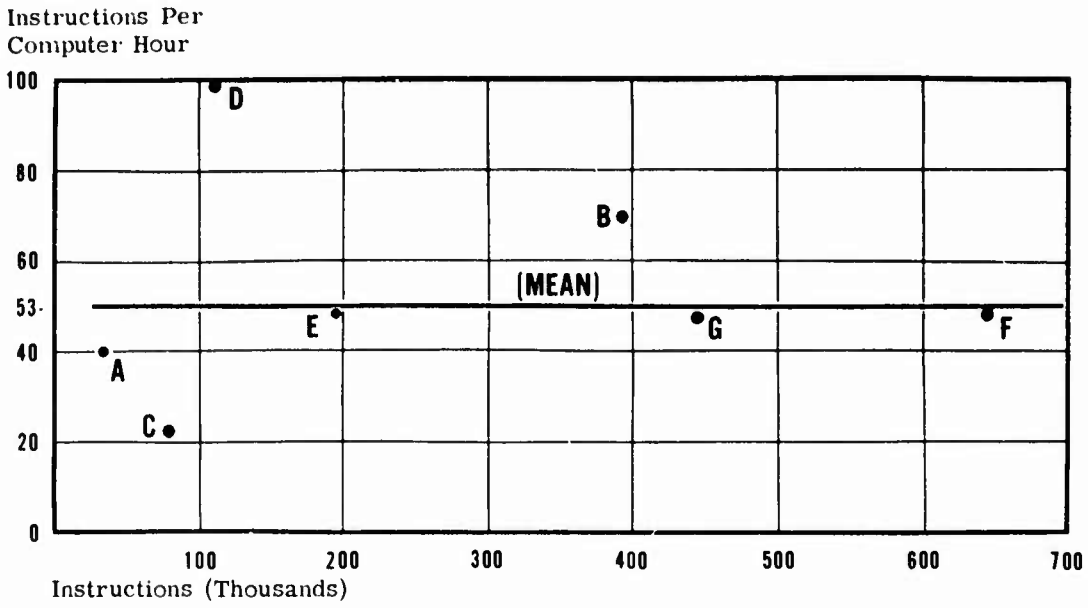


FIGURE 7. INSTRUCTIONS PER COMPUTER HOUR VERSUS PROGRAM SIZE

This figure uses the same raw data as Figure 6. The two points B and D, representing contracts that use a procedure-oriented language, fall above the mean of 53 instructions per computer hour.

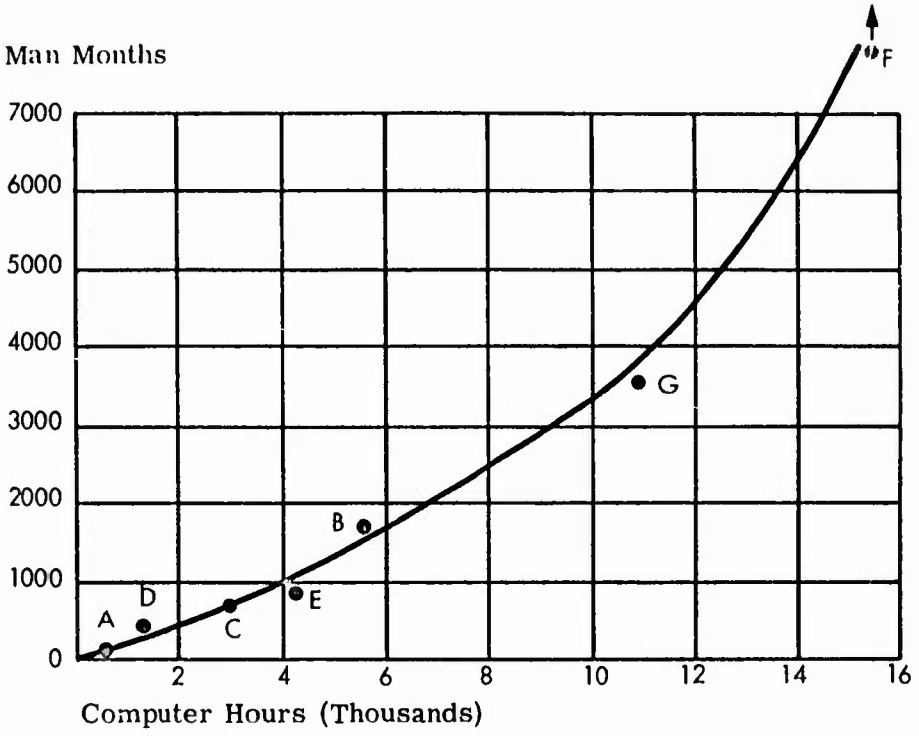


FIGURE 8. MAN MONTHS VERSUS COMPUTER HOURS

In this figure, the near-linear relationship (except for point F) between the two resources--manpower and computer time--is not surprising. Possibly, such a graph with more refined data, such as years of programmer experience as a parameter, might help managers estimate more accurately the need for these resources. Although no EAM work is included, use of "satellite" computers such as IBM's 1401 and CDC's 160 is reflected in the data. For point F, representing an incomplete contract, the computer hours are estimated to completion, but no corresponding estimate is available for the number of man months. The arrow indicates the direction the point is expected to take when the contract is completed, since additional man months will be expended.

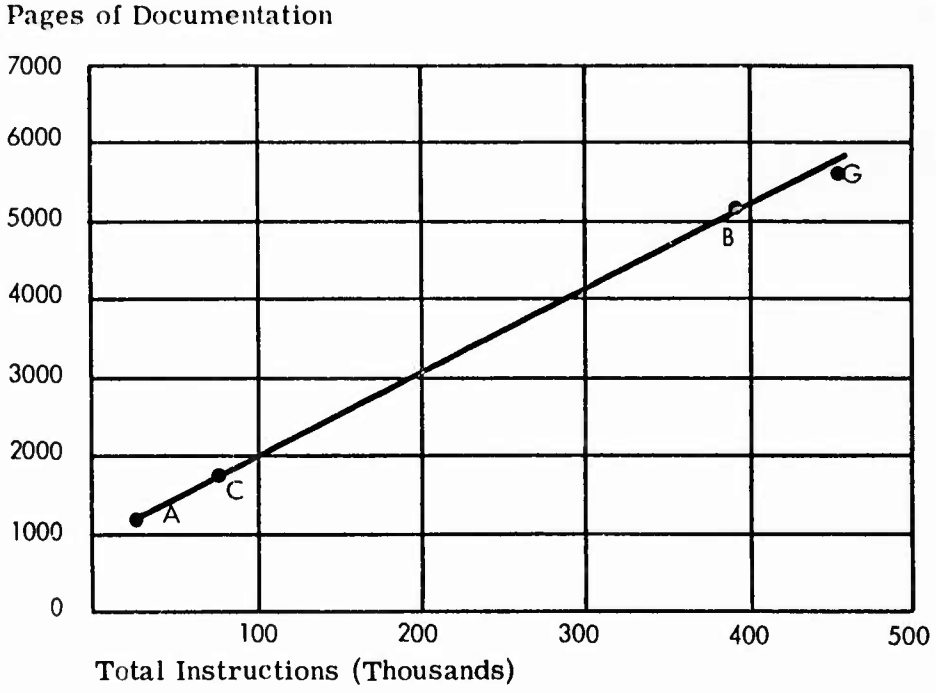


FIGURE 9. PAGES OF DOCUMENTATION VERSUS PROGRAM LENGTH

Data here are for only four of the contracts. Except for point B (an estimate of the completed task), pages of documentation refer to the number of contract pages of documentation actually produced. The graph suggests that program size in this range is linearly proportional to the number of pages of contract-required documentation; it also illustrates the recognition of documentation as a major integral part of programming for command and control systems.

TABLE 2. Contract "A" Facts

System Description

Operational System--A system for monitoring communications circuits to provide command staff support.

Program System--Program provides automatic storage and retrieval at a single center. Its basic input is teletype (automatic from outside the center) plus button queries for display information; its basic output is display information in a fixed format, e.g., geographic displays, and printouts.

Basic program functions (controlled by an executive program that permits real-time or batched inputs) are:

(1) process data to produce status, correction, and special reports
(2) record inputs on magnetic tape
(3) detect errors in inputs
(4) show status of the operational system
(5) respond to queries with hard copy and displays
(6) prepare summaries

Computer--A Philco 2000 with an 8000-word core memory; uses tape units, card reader, card punch, and high speed printer, plus a special buffer to handle teletype inputs and display outputs.

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base Size
OPS	22,339	131			
UTL	10,802	25			35,000 (48-bit) words*
TOT	33,141	156	6 months	1502	

Comments

Availability of Computer--Slippage in delivery of the operational computer required use of several remotely located substitute machines for 2/3 of testing, thus requiring additional programmer travel.

Existence of Utility Programs (at start of contract)--Philco utility programs were completed during the development and had not been completely shaken down by use.

The program system is small but logically complex. The relatively small core memory (8k) forced programmers to write programs that made very efficient use of memory.

Collection and construction of the data base consumed a large portion of the development effort.

The buffer for the teletype and display routing, plus the displays themselves, were developed concurrently with the computer program.

A small group of experienced programmers (up to 50) worked in a severe time constraint.

Procedure-oriented language was not used.

*User helped developer gather data base.

TABLE 3. Contract "B" Facts

System Description

Operational System--Operating at a single center, this system assembles data from various sensors, assesses the data, predicts and displays results; system also supports several military commands.

Program System--In semireal time, the program stores and retrieves data and, by using mathematical algorithms in a mathematical model, predicts results; the program, consisting of several systems, was developed as a series of models.

Computer--A CDC 160 (with 8,000-word memory) for I/O and a CDC 1604 (with 32,000-word memory); a TRW 400A provides color displays and a mechanism for making requests for information.

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base* Size
OPS	224,210	1426			
UTL	149,230	244			500,000 - 28
TOT	373,440	1670	12 months	5000	word entries

Comments

Availability of Computer--The computer facility was four months late, requiring additional programmer travel and inefficient testing in substitute machines; also, EAM equipment was not always available.

Existence of Utility Programs (at start of contract)--All utility programs were specially designed for this contract.

Extensive changes were made to the program requirements and also to the data base during development.

Program developer experienced much difficulty in meshing program design with two pieces of hardward--a disc file (not yet delivered) and a display console--developed concurrently.

Procedure-oriented language was used for a portion of the operational programs, but somewhat ineffectively because of a delay in its availability. Machine-oriented language was used for the utility programs.

*Data Base, as used in this document, is that part of the external environmental data of a program which exists in permanent or semipermanent storage, excluding transitory external environmental elements.

TABLE 4. Contract "C" Facts

System Description

Operational System--An experimental, real-time communications and control system. The basic input is teletype--originating both within the center and from external sources. Primary outputs are displays; also printouts and teletypes.

Program System--From planning data received by teletype, the program system computes movements and position of vehicles and displays the results to operations personnel. Program system was produced in four evolutionary packages or models.

Computer--A large general-purpose computer with a 65,000 (32-bit) word core memory and a 6-microsecond average access time. Occasionally IBM 704, 7090, and 1401 computers were used in program development to prestore card decks on tapes.

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base Size
OPS	71,400	624			
UTL	0	0			
TOT	71,400	624	26 months	1800	*

Comments

Availability of Computer--Programmers had good access to the computer; however, testing was done at two widely separated locations on machines with slightly different configurations.

Existence of Utility Programs--A full range of utility programs existed at start of contract.

Developed concurrently with the program were (1) special input equipment to handle teletype data automatically, and (2) situation-display tubes.

Not only was most of the data base available at the start of the contract, but also many of the facilities and procedures were similar to those of another contract.

This contract was characterized by informal administration, which led to uncontrolled changes and additions with consequent slippage in schedules and overruns in cost.

Procedure-oriented language was not used.

*Data base was provided to the programming contractor.

TABLE 5. Contract "D" Facts

System Description

Operational System--A vehicle tracking system that also responds to requests for data reduction.

Program System--Digitalized data from remote radars are processed to provide continuous monitoring of vehicles. Outputs are hard-copy reports of status, periodic summaries, and "flash" reports.

About 1/3 of the program is real time; the remaining 2/3 consists of data-reduction routines and mathematical processing programs. The system expands and automates a prior manual system.

Computer--Duplexed IBM 7090 with a 32,000-word core memory.

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base Size
OPS	88,654	300			
UTL	38,236	60			
TOT	126,890	360	20 months	*	**

Comments

Availability of Computer--Computer was located several hours distant from programmers, thus requiring some travel daily.

Existence of Utility Programs--A good number of the utility programs were in existence at start of contract but many had to be rewritten; also, 20,000 new utility instructions were written.

In the early phases of the contract, the lack of system requirement data hampered program development.

Procedure-oriented language was used extensively.

*Not available.
**Most of data base was in existence at start of contract.

TABLE 6. Contract "E" Facts

System Description

Operational System--A developmental effort to improve a semiautomatic system for producing training problems (e.g., enemy attack) and materials for use in exercising certain types of operator personnel.

Program System--Program performs four basic functions:
(1) updates a library of data (consisting of about 35,000 tab cards)
(2) correlates input data with problem specifications (consisting of about 30,000 tab cards)
(3) accommodates insertion of modifications to the problem simulated
(4) generates problem materials--e.g., plans, problem maps, simulated radar returns

Computer--IBM 7090 with 65,000-word core memory.

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base Size
OPS	165,128	715			
UTL	33,443	198			
TOT	198,571	913	30 months	*	**

Comments

Availability of Computer--Programmers did a large portion of their testing on alternate machines (having slightly different configurations) located in other parts of the city; also, programmers had difficulty in getting machine time on the scheduled computer.

Existence of Utility Programs--A scheduled utility system was not available per plan, requiring programmers to write many of the utility programs in three languages.

Extensive and frequent changes were made in the requirements for this system.

Procedure-oriented language was used on portions of the program.

*None contractually required.

**All of data base, consisting of 35,000 tab cards, was in existence at start of contract; however, many of the cards had to be reformatted.

TABLE 7. Contract "F" Facts

System Description

Operational System--A force-management system featuring several command centers. Each center, connected to the others by data links, contains a large computer and several automatic communications-routing devices. Each center maintains force status and plans; also, it maintains records of force movements and provides simulation capability.

Program System--Program stores and retrieves data (e.g., plans, inventories, weather); sorts data by classes, predicts movements, and prepares various hard copy outputs and displays. The program will operate at several sites.

Computer--A large solid-state general-purpose computer (single address, 48-bit word with 2.5-microsecond access time) with 65,000-word core memory plus a drum, disc, and tapes. An IBM 1401 is also used as satellite equipment.

Projected displays (in color) and high-speed printers provide the basic outputs within centers; high-speed data links send messages to automatic routing equipment (special-purpose computers).

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base Size
OPS	415,000	?			
UTL	225,000	?			
TOT	640,000	6975	40 months (Incomplete)	*	$5\frac{1}{2}$ million words

Comments

Availability of Computer--Programmers were located at a site different from that of the computer; programmers did some testing on an alternate machine in another city. Substitute machines were used--an IBM 709 and a Bendix G-15; these also were not handy to the programmers.

Existence of Utility Programs--No utility programs were in existence at start of contract. Portions of the utility programs were tested on an IBM 709 and then converted.

Program developer participated heavily in design of the operating system. The program system is highly interwoven with other subsystems.

Many changes, both large and small, were introduced into the system configuration and design.

Procedure-oriented language was much used and developed on this contract.

The contract was performed with tight security.

*Not available

(page 36 blank)

TABLE 8. Contract "G" Facts

System Description

Operational System--A real-time, man-machine system with more than 20 command centers in which environmental information is received from physically remote sensors, processed for communications and control, automatically displayed, and transmitted to remotely located operational personnel and control mechanisms.

Program System--The program, cycling continuously, stores and retrieves data, correlates the data, predicts, and prepares output messages. The particular program model surveyed consisted of four separate but compatible systems each corresponding to a command-center type. As many as 50 subprograms constitute the largest (90,000-instruction) program system. One of several successive models, this program was developed to accommodate changes in system equipment and to reduce program storage needs within the computer.

Computer--Large, duplexed computers employ 70,000-word (32 bits) core memories with average access of six microseconds. In addition to the usual card punch, card reader, printer, and tapes, input/output equipment includes displays with query buttons, data link, and teletype. (About one-half the instructions in the program are used for processing I/O data).

Numerical Data

	Number of Instructions	Number of Man Months Used	Contract Duration	Pages of Documentation	Data Base Size
OPS	286,000	3,000			
UTL	173,000	600			
TOT	459,000	3,600	36 months	5550	48,000*

Comments

Availability of Computer--Programmers had ideal access to the computer and the facility was managed by the program implementor.

Existence of Utility Programs--Almost all utility and support programs were available at start of contract from prior models of program.

Although this program model contains increased capability over earlier models, much effort was expended in making portions of programs developed for earlier models more efficient.

User delay in concurring on requirements resulted in stretch-out of program development--for comparison, an earlier model of program took two years to to develop.

Computer program will be used at a number of sites; thus some programmer travel was incurred in adapting and coordinating programs for various sites.

No procedure-oriented language was used on this contract.

*Almost all of the data base was available from prior models.

(page 38 blank)

6.0 SUMMARY

Our survey of the management problems involved in development of these seven computer programs has shown that programming managers are hampered in effectively planning and controlling tasks because they lack precise and detailed cost data, standard performance measures, and definitions of tasks and products. (The quantitative data presented in this document reflect only the cost of the program design, coding, and test activities of the programs surveyed; the only parameters collected and compared were number of program instructions, man months of effort, number of hours of computer time, and pages of documentation. Costs of operational design and of operational specifications are more difficult to compare because the tasks included in these activities lack precise definition.

We recommend, to improve program technology:

- More standardization of vocabulary to enable comparison of different programming jobs, with special emphasis on activities, resources, requirements, programming tools, program tests, documentation, budgets, and schedules;
- Research to improve definitions of activities, functions, and products and to qualify the product;
- Development of performance measures to help management plan during the course of the work and to permit comparison with other jobs;
- Collection of detailed activity and product-related cost data to serve for planning and estimating costs of current changes and new work; and
- Development of policies and procedures for applying accumulated knowledge to future programming efforts.

APPENDIX

COST FACTORS FOR ESTIMATING THE DEVELOPMENT OF LARGE PROGRAM SYSTEMS

We propose the items in the three tables that follow as significant factors in the cost of development for large program systems. Since specific quantitative relationships between these items and costs are not known, we do not indicate how cost may be estimated from these proposed factors. Also, since many or the factors listed are qualitative, exploratory analysis may be required before numerical measures can be associated with them.

The factors are divided into three groups:

1. Technical content (Table A)--The criteria for delineating the requirements for automatic data processing in the command and control system. (The same criteria probably apply as well to most large scientific and business systems.)

2. Environment (Table B)--Management elements related to development of both the command and control system and the programming job.

3. Resources (Table C)--The people and tools necessary to execute the programming job.

The factors shown in Table A are mostly beyond management control; managers can, however, by policy and practice, influence the impact on cost of the items in Tables B and C.

TABLE A

Technical Content

(Requirements)

Increases in the following requirements for any particular command and control system generally tend to increase the cost of program development.

1. Number of command centers in the system and, in particular, the number that will use automatic data processing.

2. Number of different computers for which programs are prepared.

3. Remoteness and separation (from one another and from the location of the program development) of centers containing computers.

4. Number of interfaces--number of unique types of data input and output, to and from the command and control system, and more specifically, to and from the computer(s).

5. Frequency of inputs and outputs to the command and control system and the computer(s).

6. Number of equipment components (related to the data processing) being developed concurrently with the program (particularly the computer and input/output devices).

7. Number and diversity of the information-processing functions--particularly those proposed for automatic data processing--in the command and control system.

8. Number of different program systems of the three general types--operational, utility, and support.

9. Number of entries (total size) for the data base and the number of different types of data needed for it.

10. Speed of response for the automatic data processing.

11. Speed of recovery for automatic data processing from malfunction or failure.

12. Complexity of the data-processing functions. For example, the data to be processed could be scored by assigning measures (or weights reflecting increased complexity) to the following:

Class	Functions
Clerical	store, retrieve, reformat
Synoptic	reduce, classify
Predictive	predict, forecast based upon pre-established performance criteria
Directive	decide, based upon pre-established decision criteria

and summing the product of class weight times the number of functions to be performed in each class.

13. Interdependence of the data-processing functions. For example, each function could be rated according to the following criteria:

    a. Number of types of data that are not raw inputs--i.e., data that must be preprocessed.

    b. Number of functions that use the same preprocessed data.

    c. Number of data types that are used by many functions--i.e., that are usually stored in a central table in the computer.

14. Number of changes, particularly those made after program design work, has begun.

15. Degree of innovation in the system, its components, and especially the automatic-data-processing function.

16. Security level of the system and the information.

TABLE B*

Environment

(Management)

Increases in the following factors, or in their efficacy, should reduce program-development costs.

1. The use for system development of a management plan that (a) recognizes the relation of program development to system development, (b) delineates responsibility and authority (as they relate to objectives) among the component developers, (c) specifies communications and decision-making procedures, and (d) includes within the plan a mechanism for changing it.

2. The extent to which this management plan is maintained and monitored.

3. The use, maintenance, and monitoring of a similar management plan within the program developer's organization.

4. The use of reasonable schedules

5. The extent to which the program can be developed by a small group of people.

6. The extent to which the customer has clarified his objectives and stated his requirements.

7. The extent to which a well-known, acceptable procedure exists for considering, executing, scheduling, and costing changes.

8. Stability of requirements during the production activities in program development.

9. The extent to which timely, well-written technical plans exist and are readily available for both system and program development.

10. The extent to which policies are formulated and applied for documentation and quality control during program development.

---

*The cost elements in this table reflect the heavy emphasis in command and control systems on coordination and participation of many organizations and large numbers of people.

TABLE C

Resources

The factors below are resources available to the program developer for executing his job. Any increase in these will probably reduce the development cost.

1. Experience level of personnel in computer programming.
2. Availability of the computer.
3. Speed of response time for the computer facility.
4. Proficiency of computer operators.
5. Experience level of the military personnel involved.
6. Existence of various programs such as compilers, assemblers, and utility programs, library routines.
7. Computer capability (storage capacity, operate time, and the power of its order code).

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
MANAGEMENT OF COMPUTER PROGRAMMING
FOR COMMAND AND CONTROL SYSTEMS -
A SURVEY.
Scientific rept., TM-903/000/02,
by N. Claussen, K. Heinze, V. LaBolle.
8 May 1963, 43p., 8 tables.
(Contract SD-97)

Unclassified report

DESCRIPTORS: Command Systems and
Control Systems. Programming
(Computers).

UNCLASSIFIED

Supersedes TM-903/000/01. Reports findings of a survey of seven computer programming efforts for command and control systems. Considers the characteristics of computer programming, including its growth, military use, and personnel. Discusses management difficulties such as unavailability of a computer and the effects of customer delay in design decisions. Concludes that knowledge of managing and developing computer programming systems must be extended and detailed, and that programming must be formalized.

UNCLASSIFIED

UNCLASSIFIED